Two iterative methods for solving $Ax = b$

Raymond Khoa, jckhoa@yahoo.com, 30 April 2013

***Abstract*—A linear algebra system is represented in matrix form as $Ax = b$ where A is a given matrix, b a given vector and x the unknown variable to be found. Steepest Descent and Conjugate Gradient are well known iterative methods to solve this system. These methods require matrix A to be symmetric and positive-definite. This article introduces two iterative methods: Method 1 and Method 2 which bear resemblances to the aforementioned two methods while not requiring A to be symmetric nor positive-definite.**

***Index Terms*—steepest descend, conjugate gradient, iterative method, linear algebra system, nonsymmetric, nonpositive-definite**

I. Iterative methods

Iterative methods aim to solve $Ax = b$ by a sequenc of guesses. The new guess of the solution x_n usually depends on the last guess x_{n-1}

$$x_n = f(x_{n-1})$$

where $f(..)$ is the rule for guessing

The guessing process stops when $Ax_n \simeq b$ or the residual $|r_n| = |b - Ax_n| \simeq 0$

The key to iterative methods is to formulate $f(..)$ such that the guesses **converge quickly** to the solution

II. Method 1

The guessing rule for Method 1 is

$$x_n = f(x_{n-1}) = x_{n-1} + \alpha_{n-1} r_{n-1}$$

i.e. the new guess x_n is dependent on the last guess x_{n-1} and the last residual r_{n-1} . Here, α_{n-1} is a scalar

We start with the initial guess x_0. Its residual $r_0 = b - Ax_0$

The next guess, by the rule, $x_1 = x_0 + \alpha_0 r_0$

What is the value of α_0 that best approximates x_1? In other words, what value of α_0 can minimize the residual $|r_1|$ incurred by x_1?

$$r_1 = b - Ax_1 = b - A(x_0 + \alpha_0 r_0) = r_0 - \alpha_0 A r_0$$

$$|r_1| = |r_0 - \alpha_0 A r_0| = |\alpha_0 A r_0 - r_0|$$

Minimizing $|r_1|$ is equivalent to minimizing $|\alpha_0 A r_0 - r_0|$

FigureII.1 uses vector diagram to show the possible position for $\alpha_0 A r_0 - r_0$

It is observed in the figure that $|\alpha_0 A r_0 - r_0|$ has its smallest value when $\alpha_0 A r_0 - r_0$ is orthogonal to Ar_0, i.e.

$$(Ar_0)^T(\alpha_0 A r_0 - r_0) = 0$$

$$\alpha_0 = \frac{(Ar_0)^T r_0}{(Ar_0)^T (Ar_0)}$$

Hence, $\alpha_{n-1} = \frac{(Ar_{n-1})^T r_{n-1}}{(Ar_{n-1})^T (Ar_{n-1})}$

Method 1 is summarized in the box below

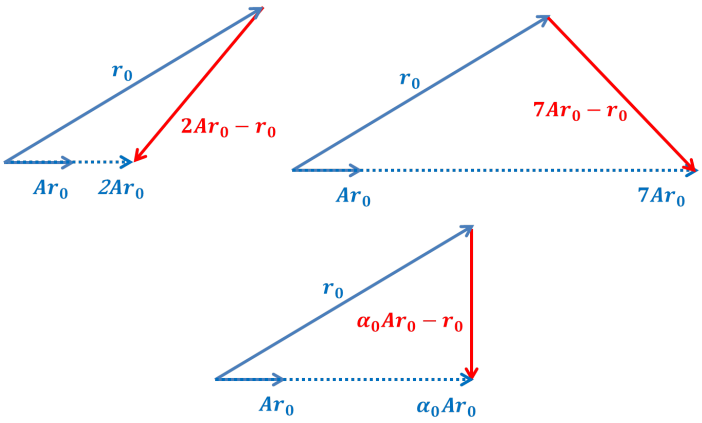

Figure II.1. Illustration of the best position for $\alpha_0 A r_0 - r_0$ by vector diagram

1) Start with the first guess x_0 and calculate the first residual
   $r_0 = b - x_0$
2) Repeat until $|r_n| \simeq 0$ or a certain number of iterations is reached
   $\alpha_{n-1} = \frac{(Ar_{n-1})^T r_{n-1}}{(Ar_{n-1})^T (Ar_{n-1})}$
   $x_n = x_{n-1} + \alpha_{n-1} r_{n-1}$
   $r_n = b - Ax_n$

III. Method 2

A. Guessing rule:

Method 2 uses this guessing rule:

$$x_n = x_{n-1} + \alpha_{n-1} d_{n-1} \quad (1)$$

The new guess is no more dependent on the last residual but rather on a certain vector d_{n-1}. Our aim is to find the best α_{n-1} and d_{n-1} to minimize $|r_n|$

$$r_n = b - Ax_n = b - A(x_{n-1} + \alpha_{n-1} d_{n-1})$$

$$r_n = r_{n-1} - \alpha_{n-1} A d_{n-1} \quad (2)$$

Moreover, $r_{n-1} = r_{n-2} - \alpha_{n-2} A d_{n-2}$, and so on,

$$r_n = r_{n-2} - \alpha_{n-2} A d_{n-2} - \alpha_{n-1} A d_{n-1} = ...$$

$$r_n = r_0 - \sum_{i=0}^{n-1} \alpha_i A d_i \quad (3)$$

B. Best value of α_i to minimize $|r_n|$

Mimizing $|r_n|$ is equivalent to minimizing $|r_n|^2$

$$|r_n|^2 = (r_0 - \sum_{i=0}^{n-1} \alpha_i A d_i)^T (r_0 - \sum_{i=0}^{n-1} \alpha_i A d_i)$$

$$= r_0^2 - 2\sum_{i=0}^{n-1} \alpha_i (Ad_i)^T r_0 + \sum_{j=0}^{n-1} \sum_{i=0}^{n-1} \alpha_j \alpha_i (Ad_j)^T A d_i \quad (4)$$

The term $\sum_{i=0}^{n-1} \alpha_i A d_i$ in (3) is a linear combination of Ad_i. Let us assume that all Ad_i vectors are orthogonal to each other so that (4) is simplified to

$$|r_n|^2 = r_0^2 - 2\sum_{i=0}^{n-1} \alpha_i (Ad_i)^T r_0 + \sum_{i=0}^{n-1} \alpha_i^2 (Ad_i)^T A d_i$$

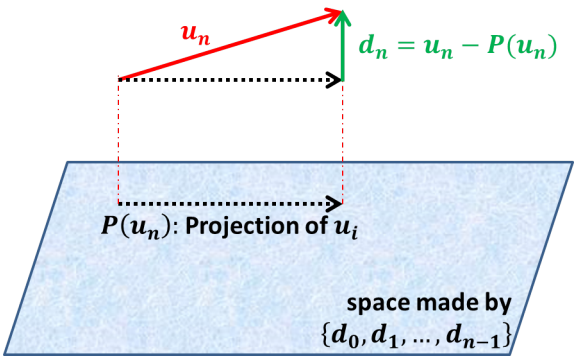

Figure III.1. Illustration of Gram Schmidt method

By Larange's multiplier, minimizing $|r_n|^2$ requires

$$\frac{\partial}{\partial \alpha_i}|r_n|^2 = 0$$
$$-2(Ad_i)^T r_0 + 2\alpha_i (Ad_i)^T Ad_i = 0$$
$$\alpha_i = \frac{(Ad_i)^T r_0}{(Ad_i)^T Ad_i} \quad (5)$$

C. The residual r_n is orthogonal to previous Ad_i

Multiply both sides of (3) by $(Ad_j)^T$ we have

$$(Ad_j)^T r_n = (Ad_j)^T r_0 - \sum_{i=0}^{n-1} \alpha_i (Ad_j)^T Ad_i \quad (6)$$

- if $j < n$, since Ad_i are orthogonal to each other, (6) is simplified to
$$(Ad_j)^T r_n = (Ad_j)^T r_0 - \alpha_j (Ad_j)^T Ad_j \quad (7)$$
substituting (5) into (7) for α_j, we obtain
$$(Ad_j)^T r_n = (Ad_j)^T r_0 - \frac{(Ad_j)^T r_0}{(Ad_j)^T Ad_j}(Ad_j)^T Ad_j = 0$$
Hence **the residual is orthogonal to all previous** Ad_i, i.e. $r_n^T (Ad_i) = 0, \forall i < n$
- if $j \geq n$, since Ad_i are orthogonal to each other, (6) is simplified to
$$(Ad_j)^T r_n = (Ad_j)^T r_0$$
Hence, when $j = n$, $(Ad_j)^T r_j = (Ad_j)^T r_0$, and we can rewrite (5) as such
$$\alpha_j = \frac{(Ad_j)^T r_j}{(Ad_j)^T Ad_j} \quad (8)$$

D. Finding d_n

All d_n must satisfy the condition that Ad_n are orthogonal to each other. We can make use of Gram Schmidt method to achieve this.

The Gram Schmidt method generates m orthogonal vectors d_n from m given arbitrary vectors u_n. The method is as follow

1) $d_0 = u_0$
2) For $n = 1$ to $m - 1$
   $d_n = u_n - \sum_{i=0}^{n-1} \beta_i d_i$ where $\beta_i = \frac{{u_n}^T d_i}{d_i^T d_i}$

Figure III.1 illustrates how Gram Schmidt method works. In the figure, the space made by previous d_i is represented by a plane. Subtracting the projection of u_n from itself gives a vector orthogonal to the plane.

The Gram Schmidt method only generates d_n such that d_n are orthogonal to each other while we want to generate d_n such that Ad_n are orthogonal to each other.

We make use of the rule $d_n = u_n - \sum_{i=0}^{n-1} \beta_i d_i$ from Gram Schmidt method to achieve the purpose.

Multiplying both sides of this rule by matrix A gives

$$Ad_n = Au_n - \sum_{i=0}^{n-1} \beta_i Ad_i$$

Multiplying both sides of the last equation by $(Ad_i)^T$ with $i < n$ and using the fact that Ad_i are orthogonal to each other, we obtain

$$0 = (Ad_i)^T (Au_n) - \beta_i (Ad_n)^T (Ad_n)$$
$$\beta_i = \frac{(Ad_i)^T (Au_n)}{(Ad_i)^T (Ad_i)}$$

So here is the method to generate d_n such that Ad_n are orthogonal to each other from given sets of vectors u_n

1) $d_0 = u_0$
2) For $n = 1$ to $m - 1$
   $d_n = u_n - \sum_{i=0}^{n-1} \frac{(Ad_i)^T (Au_n)}{(Ad_i)^T (Ad_i)} d_i$

E. Method 2 with given u_n

If a set of vectors u_n is given, method 2 is as follows

1) Start with the first guess x_0 and calculate the first residual $r_0 = b - Ax_0$
2) $d_0 = u_0$
3) Repeat until $|r_n| \simeq 0$ or a certain number of iterations is reached
   $\alpha_{n-1} = \frac{(Ad_{n-1})^T r_{n-1}}{(Ad_{n-1})^T (Ad_{n-1})}$
   $x_n = x_{n-1} + \alpha_{n-1} d_{n-1}$ (guessing rule for method 2)
   $r_n = r_{n-1} - \alpha_{n-1} (Ad_{n-1})$ (result from (2))
   $d_n = u_n - \sum_{i=0}^{n-1} \frac{(Ad_i)^T (Au_n)}{(Ad_i)^T (Ad_i)} d_i$

F. Convergence of method 2:

Assume that matrix A is $m \times m$. Hence, the dimention of A and the space of Ax is at most m. In Method 2, the vectors Ad_i are orthogonal basis of Ax. Therefore, there are at most m non-zero vectors Ad_i

If r_0 belongs to the space Ax, i.e. r_0 can be expressed as a linear combination of Ad_i,

$$r_0 = \sum_{i=0}^{m-1} \gamma_i Ad_i \quad (9)$$

To find γ_i, multiply both sides of the last equation by $(Ad_i)^T$ and using the fact that Ad_i are orthogonal to each other to obtain

$$(Ad_i)^T r_0 = \gamma_i (Ad_i)^T (Ad_i)$$
$$\gamma_i = \frac{(Ad_i)^T r_0}{(Ad_i)^T (Ad_i)}$$

Substituting the result for γ_i into (9), we have

$$r_0 = \sum_{i=0}^{m-1} \frac{(Ad_i)^T r_0}{(Ad_i)^T (Ad_i)} Ad_i \quad (10)$$

From (3), after m iterations we have

$$r_m = r_0 - \sum_{i=0}^{m-1} \alpha_i Ad_i \quad (11)$$

Substituting the result for r_0 from (10) and α_i from (5) into (11), we have

$$r_m = \sum_{i=0}^{m-1} \frac{(Ad_i)^T r_0}{(Ad_i)^T (Ad_i)} Ad_i - \sum_{i=0}^{m-1} \frac{(Ad_i)^T r_0}{(Ad_i)^T (Ad_i)} Ad_i = 0$$

=> Method 2 gives a solution after m iterations if r_0 lies in the space Ax

Even if r_0 does not lie in the space Ax, after m iteration, the solution will converge to a value such that $|b - Ax|$ is minimized.

G. Choice of u_i

$u_i$can be chosen arbitrarily. However, if not chosen carefully, for example, one accidentally chooses u_i dependent on previous d_i, one will just waste one iteration because d_i genernated from that u_i will be 0.

One good choice of u_i is r_i, i.e., set $u_i = r_i$ for each iteration for the reason that if r_i is zero at a certain step, we need to iterate no more because the solution is already reached.

So here is Method 2 with $u_i = r_i$

1) Start with the first guess x_0 and calculate the first residual r_0
2) $d_0 = r_0$
3) Repeat until $|r_n| \simeq 0$ or a certain number of iterations is reached
   $\alpha_{n-1} = \frac{(Ad_{n-1})^T r_{n-1}}{(Ad_{n-1})^T (Ad_{n-1})}$
   $x_n = x_{n-1} + \alpha_{n-1} d_{n-1}$
   $r_n = r_{n-1} - \alpha_{n-1} (Ad_{n-1})$
   $d_n = r_n - \sum_{i=0}^{n-1} \frac{(Ad_i)^T (Ar_n)}{(Ad_i)^T (Ad_i)} d_i$ (12)

Still, Method 2 with $u_i = r_i$ has a shortcoming that we need to remember previous d_i after each step (see equation (12)) and the summation in (12) is computational comsuming.

Is it possible to eliminate the summation?

H. Eliminate the summation in (12)

From (2), we have

$$r_{i+1} = r_i - \alpha_i A d_i \text{ => } Ad_i = \frac{r_i - r_{i+1}}{\alpha_i}$$

Substituting this result for $(Ad_i)^T$ in (12), we obtain

$$d_n = u_n - \sum_{i=0}^{n-1} \frac{\left(\frac{r_i - r_{i+1}}{\alpha_i}\right)^T (Au_n)}{(Ad_i)^T (Ad_i)} d_i$$

$$d_n = u_n - \sum_{i=0}^{n-1} \frac{r_i^T (Au_n)}{\alpha_i (Ad_i)^T (Ad_i)} d_i + \sum_{i=0}^{n-1} \frac{r_{i+1}^T (Au_n)}{\alpha_i (Ad_i)^T (Ad_i)} d_i \text{ (13)}$$

If Au_n is orthogonal to previous r_i, i.e. $r_i^T (Au_n) = 0 \; \forall i < n$, (13) is simplified to

$$d_n = u_n + \frac{r_n^T (Au_n)}{\alpha_{n-1} (Ad_{n-1})^T (Ad_{n-1})} d_{n-1}$$

Substituting the value for α_{n-1} from (8) into the last equation and simplify it to obtain

$$d_n = u_n + \frac{r_n^T (Au_n)}{(Ad_{n-1})^T r_{n-1}} d_{n-1} \text{ (14)}$$

As seen in (14), the summation is gone. Therefore, our job is to choose u_n such that

$\mathbf{Au_n}$ is orthogonal to previous $\mathbf{r_i}$ (i.e. $\forall \mathbf{i < n}$) (15)

From (12), u_n can be expressed as

$$u_n = d_n + \sum_{i=0}^{n-1} \beta_i d_i \text{ where } \beta_i = \frac{(Ad_i)^T (Au_n)}{(Ad_i)^T (Ad_i)}$$

$$Au_n = Ad_n + \sum_{i=0}^{n-1} \beta_i A d_i$$

In other words, Au_n is a linear combination of $Ad_i \; \forall i \leq n$

We know from III-C that r_n is orthogonal to previous Ad_i (i.e. for $\forall i < n$). Hence, r_n is orthogonal to any linear combination of $Ad_i \; \forall i < n$

Therefore, because Au_n is a linear combination of $Ad_i \forall i \leq n$,

$$r_n \text{ is orthogonal to previous } Au_i \text{ (i.e.} \forall i < n) \text{ (16)}$$

A possible choice of u_n to satisfy (15) is $u_n = A^T r_n$. Because then we have

$$\text{(16) <=> } r_n^T (Au_i) = 0 \text{ for } \forall i < n$$

$$\text{<=> } \left(A^T r_n\right)^T u_i = 0 \text{ for } \forall i < n$$

$$\text{<=> } u_n^T u_i = 0 \text{ for } \forall i < n \text{ (using } u_n = A^T r_n \text{)}$$

$$\text{<=> } u_n^T \left(A^T r_i\right) = 0 \text{ for } \forall i < n \text{ (using } u_i = A^T r_i \text{)}$$

$$\text{<=> } (Au_n)^T r_i = 0 \text{ for } \forall i < n$$

<=> Au_n is orthogonal to previous r_i <=> (15)

We rewrite (14) when $u_n = A^T r_n$ as follows

$$d_n = A^T r_n + \frac{r_n^T \left[A\left(A^T r_n\right)\right]}{r_{n-1}^T (Ad_{n-1})} d_{n-1}$$

$$d_n = A^T r_n + \frac{\left(A^T r_n\right)^T \left(A^T r_n\right)}{r_{n-1}^T (Ad_{n-1})} d_{n-1}$$

Here is Method 2 with $u_n = A^T r_n$

1) Start with the first guess x_0 and calculate the first residual $r_0 = b - Ax_0$
2) $d_0 = A^T r_0$
3) Repeat until $|r_n| \simeq 0$ or a certain number of iterations is reached
   $\alpha_{n-1} = \frac{(Ad_{n-1})^T r_{n-1}}{(Ad_{n-1})^T (Ad_{n-1})}$
   $x_n = x_{n-1} + \alpha_{n-1} d_{n-1}$
   $r_n = r_{n-1} - \alpha_{n-1} (Ad_{n-1})$
   $d_n = A^T r_n + \frac{\left(A^T r_n\right)^T \left(A^T r_n\right)}{(Ad_{n-1})^T r_{n-1}} d_{n-1}$

IV. Contrast of Method 1 vs Steepest Descent Method and Method 2 vs Conjugate Gradient Method

Following are two tables showing the contrast of **Method 1 vs Steepest Descent Method** and **Method 2 vs Conjugate Gradient Method**

A. Steepest Descent Method vs Method 1

Steepest Descent Method [1]	Method 1
1) Start with the first guess x_0 and calculate the residual $r_0 = b - Ax_0$ 2) Repeat until $\|r_n\| \simeq 0$ or a certain number of iterations is reached $\alpha_{n-1} = \frac{r_{n-1}^T r_{n-1}}{r_{n-1}^T (Ar_{n-1})}$ $x_n = x_{n-1} + \alpha_{n-1} r_{n-1}$ $r_n = b - Ax_n$	1) Start with the first guess x_0 and calculatethe residual $r_0 = b - Ax_0$ 2) Repeat until $\|r_n\| \simeq 0$ or a certain number of iterations is reached $\alpha_{n-1} = \frac{(Ar_{n-1})^T r_{n-1}}{(Ar_{n-1})^T (Ar_{n-1})}$ $x_n = x_{n-1} + \alpha_{n-1} r_{n-1}$ $r_n = b - Ax_n$
Requirements: Matrix A has to be symmetric and positive definite	Requirements: not any

B. Conjugate Gradient Method vs Method 2

Conjugate Gradient Method [1]	Method 2
1) Start with the first guess x_0 and calculate the first residual $r_0 = b - Ax_0$ 2) $d_0 = r_0$ 3) Repeat until $\|r_n\| \simeq 0$ or a certain number of iterations is reached $\alpha_{n-1} = \frac{r_{n-1}^T r_{n-1}}{d_{n-1}^T (Ad_{n-1})}$ $x_n = x_{n-1} + \alpha_{n-1} d_{n-1}$ $r_n = r_{n-1} - \alpha_{n-1} (Ad_{n-1})$ $d_n = r_n + \frac{r_n^T r_n}{r_{n-1}^T r_{n-1}} d_{n-1}$	1) Start with the first guess x_0 and calculate the first residual $r_0 = b - Ax_0$ 2) $d_0 = A^T r_0$ 3) Repeat until $\|r_n\| \simeq 0$ or a certain number of iterations is reached $\alpha_{n-1} = \frac{(Ad_{n-1})^T r_{n-1}}{(Ad_{n-1})^T (Ad_{n-1})}$ $x_n = x_{n-1} + \alpha_{n-1} d_{n-1}$ $r_n = r_{n-1} - \alpha_{n-1} (Ad_{n-1})$ $d_n = A^T r_n + \frac{(A^T r_n)^T (A^T r_n)}{(Ad_{n-1})^T r_{n-1}} d_{n-1}$
Requirements: Matrix A has to be symmetric and positive definite	Requirements: not any

V. Implementation

Matlab implementation can be found on the author's blog [2]

VI. Conclusion

This article presented the mathematical background of two iterative methods: Method 1 and Method 2. Both methods can be used for any square matrix A to solve $Ax = b$. Theoretically, Method 2 gives the solution after at most m iterations where m is the size of matrix A.

References

[1] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," August 1994. [Online]. Available: http://www.cs.cmu.edu/ quake-papers/painless-conjugate-gradient.pdf

[2] R. Khoa, "Conjugate gradient method for nonsymmetric and nonpositive definite matrices," April 2013. [Online]. Available: http://sharewithraymond.blogspot.de/2013/04/conjugate-gradient-method-for.html